(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 May 2006 (04.05.2006)

PCT

(10) International Publication Number
WO 2006/047092 A2

(51) International Patent Classification:
H04L 12/413 (2006.01) *H04J 3/24* (2006.01)

(21) International Application Number:
PCT/US2005/036700

(22) International Filing Date: 13 October 2005 (13.10.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/621,396 22 October 2004 (22.10.2004) US
11/155,388 16 June 2005 (16.06.2005) US

(71) Applicant *(for all designated States except US)*: **CISCO TECHNOLOGY, INC.** [US/US]; 170 West Tasman Drive, San Jose, CA 95134 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BERGAMASCO, Davide** [IT/US]; 376 Flora Vista Avenue, Sunnyvale, CA 94086 (US). **BONOMI, Flavio** [IT/US]; 526 Lowell Avenue, Palo Alto, CA 94301 (US). **ALARIA, Valentina** [IT/US]; 14 Pleasant Street, San Francisco, CA 94108 (US). **BALDINI, Andrea** [IT/US]; 4153 23rd Street, San Francisco, CA 94114 (US).

(74) Agents: **SAMPSON, Roger, S.** et al.; Beyer Weaver & Thomas, LLP, P.O. Box 70250, Oakland, CA 94612-0250 (US).

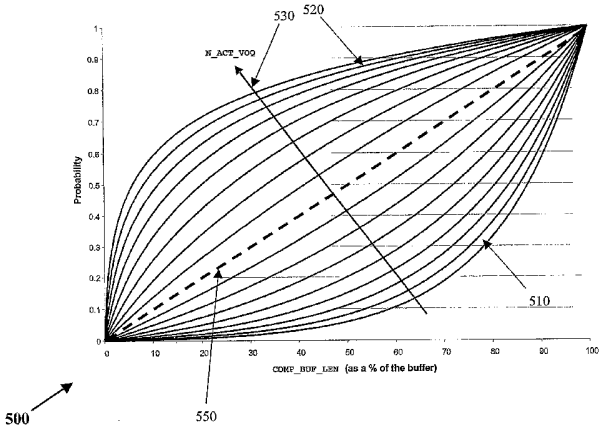
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) **Title:** ACTIVE QUEUE MANAGEMENT METHODS AND DEVICES



(57) **Abstract:** Novel methods and devices are provided for AQM of input-buffered network devices. Preferred implementations of the invention control overall buffer occupancy while protecting uncongested individual VOQs. The probability of setting a "global drop flag" (which is not necessarily used to trigger packet drops, but may also be used to trigger other AQM responses) may depend, at least in part, on the lesser of a running average of buffer occupancy and instantaneous buffer occupancy. In some preferred embodiments, this probability also depends on the number of active VOQs. Moreover, a global drop flag is set in conjunction with a drop threshold *M* associated with the VOQs. Whether an AQM response is made may depend on whether a global drop flag has been set and whether a destination VOQ contains *M* or more packets. Different*M* values may be established for different classes of traffic, e.g., with higher M values for higher-priority traffic. AQM responses (e.g., to drop packets) may be taken more aggressively when there is a larger number of active VOQs.

WO 2006/047092 A2

ACTIVE QUEUE MANAGEMENT METHODS AND DEVICES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to United States Provisional Application No. 60/621,396 (Attorney Docket No. CISCP404P), entitled "FC Over Ethernet" and filed on October 22, 2004, and United States Patent Application No. 11/155,388 (Attorney Docket No: CISCP439), entitled "Active Queue Management Methods And Devices" and filed on June 16, 2005, which are incorporated by reference for all purposes.

BACKGROUND OF THE INVENTION

Congestion avoidance techniques are essential to the operation of networks and network devices. One such technique known in the art as "Random Early Discard" or "RED" is described in a publication by S. Floyd and V. Jacobson entitled *"Random Early Detection Gateways for Congestion Avoidance,"* (Transactions on Networking, August 1993), which is hereby incorporated by reference for all purposes.

The basic principle behind RED is to control the average length of a network device's (e.g., a router's) output queue in order to avoid long term congestion. To achieve this goal, RED must work tightly coupled with transport protocols, such as TCP, which are equipped with their own congestion avoidance mechanisms and are thus capable to react to congestion indications generated by RED routers.

Fig. 1 includes graph 100 that illustrates how RED works. For each incoming packet, the average queue length is calculated. (Please note that the terms "packet" and "frame" may be used interchangeably herein.) If the average queue length is below a predefined minimum threshold 105, the packet is accepted and stored in the output queue for transmission. If the average queue size is above the minimum threshold 105 but below a predefined maximum threshold 110, a packet marking probability is computed and the packet gets marked according to this probability. The marking probability is proportional to the average queue size. Therefore, when the queue is larger, there is a higher probability for an incoming packet to get marked. Finally, if the average queue size is above the maximum threshold 110, all incoming

packets are marked until the average queue size falls again below the maximum threshold 110.

It is responsibility of the transport protocol to take the appropriate countermeasures when it detects packets marked by RED. When TCP is being used in the absence of an explicit method of marking packets, packets can only be "marked" by discarding them, with TCP interpreting the loss of packets as a congestion indication. When packet drops are detected, TCP sources immediately reduce their transmission rate, causing a reduction of the traffic volume at the congested router(s). Discarding packets is also a useful means to control average queue size when non-reactive transport protocols such as UDP are exploited. The RED algorithm has been implemented in some output-buffered network devices, including the Catalyst 6500 switch and the 7200 series router provided by Cisco Systems, Inc., to prevent congestion of output buffers.

It would seem to be desirable to deploy an active queue management ("AQM") technique such as RED in input-buffered switches to keep congestion of *input buffers and Virtual Output Queues ("VOQs") under control. However, the* RED algorithm presents scalability issues and other challenges. In an output-buffered network device having N ports, at least N instances of the RED algorithm need to be applied. It may not be feasible to have a dedicated instance of the RED mechanism associated with each VOQ of an input-buffered network device. In such a device, there could be thousands of VOQs mapped on a physical buffer. Because each input buffer implements a VOQ for each output of the system, N^2 instances of the RED algorithm would need to be deployed for an input-buffered switch having N ports. It would be both difficult and expensive to provide an average queue length estimator to each VOQ.

If one were to apply a traditional RED algorithm on an entire buffer shared by many VOQs, flows that are not contributing to congestion could be adversely affected. On the other hand, if a traditional RED method were applied on a per-VOQ basis for potentially thousands of VOQs, each queue could be too small to obtain a meaningful average occupancy value, particularly if the buffer is small with respect to the number of VOQs.

Considering the foregoing, a traditional RED method would not seem to be a desirable way to implement AQM for input-buffered switches. It would be desirable to implement methods and devices that overcome at least some of the aforementioned shortcomings of the prior art.

SUMMARY OF THE INVENTION

Novel methods and devices are provided for AQM of input-buffered network devices. Preferred implementations of the invention control overall buffer occupancy while protecting uncongested individual VOQs.

According to some aspects of the invention, the probability of setting a "global drop flag" depends, at least in part, on the lesser of a running average of buffer occupancy and instantaneous buffer occupancy. As noted elsewhere herein, a "drop flag" is not necessarily used to trigger packet drops, but may also be used to trigger other AQM responses. Similarly, the term "drop probability" and the like do not necessarily relate to dropping packets, but can be used with regard to other types of AQM responses. In some preferred embodiments, this probability also depends on the number of active VOQs.

Moreover, a global drop flag is preferably set in conjunction with a drop threshold M associated with the VOQs. Whether an AQM response is made preferably depends on both whether a global drop flag has been set and whether a destination VOQ contains M or more packets. Some implementations set different M values for different classes of traffic, e.g., with higher M values for higher-priority traffic. AQM responses (e.g., to drop packets) may be taken more aggressively when there is a larger number of VOQs. Therefore, M is preferably smaller when there is a larger number of active VOQs.

Some implementations of the invention provide an active queue management ("AQM") method that includes these steps: receiving an incoming packet; estimating an overall occupancy level O of an input buffer; determining a number N of active virtual output queues ("VOQs") of the input buffer; computing a probability P based on the overall occupancy level and the number of VOQs; generating a random number R; and setting a flag when R is less than P. Preferably, P increases when N increases.

In some such implementations, *O* is a running average occupancy level of the input buffer. In other implementations, *O* is the lesser of *R* and *I,* wherein *R* is a running average occupancy level of the input buffer and *I* is an instantaneous occupancy level of the input buffer.

The method may also include these steps: determining an AQM threshold *M* based, at least in part, on *N;* determining a destination VOQ for the packet; and determining whether the destination VOQ contains *M* or more packets. *M* preferably decreases as the number of active VOQs increases. The packet may be forwarded to the destination VOQ when the destination VOQ contains fewer than *M* packets. When the destination VOQ contains *M* or more packets, an AQM action may be taken. The AQM action may involve dropping the packet, dropping a packet at the head of the destination VOQ, marking the packet and/or sending an explicit congestion notification.

Some aspects of the method involve the following steps: determining a plurality of AQM thresholds, each AQM threshold being based at least in part on a packet's class of service, determining a class of service of the packet; determining a destination VOQ for the packet; and determining whether the destination VOQ contains more packets than the AQM threshold for the determined class of service. An AQM action may be taken when the destination VOQ contains more packets than the AQM threshold for the determined class of service.

Alternative implementations of the invention provide another AQM method, comprising these steps: receiving an incoming packet; estimating an overall occupancy level *O* of an input buffer; computing a probability *P* based at least in part on the overall occupancy level; generating a random number *R*; setting a flag if *R* is less than *P*; determining an AQM threshold *M;* and determining whether a destination VOQ of the packet contains *M* or more packets. The packet may be sent to the destination VOQ when the destination VOQ contains fewer than *M* packets. An AQM action may be taken when the destination VOQ contains *M* or more packets. The AQM action may involve dropping the packet, dropping a packet at the head of

the destination VOQ, marking the packet and/or sending an explicit congestion notification.

P may also be based in part upon a number of active VOQs. *P* preferably increases when the number of VOQs increases. *O* may be a running average occupancy level of the input buffer. *O* may also be the lesser of *R* and *I*, wherein *R* is a running average occupancy level of the input buffer and *I* is an instantaneous occupancy level of the input buffer. *M* preferably decreases as the number of active VOQs increases.

Alternative methods of the invention also involve managing input buffers in a network device. One such method includes these steps: determining a first range of values of *N*, where *N* is the number of active VOQs in an input buffer; determining a second range of values of *N*; assigning a first family of probabilistic drop functions to the first range of values; and assigning a second family of probabilistic drop functions to the second range of values. The first family of probabilistic drop functions are exponential functions of *N* and computed buffer occupancy *C*, whereas the second family of probabilistic drop functions are logarithmic functions of *N* and *C*. The method also includes the steps of determining *N* at a first time, determining a first probabilistic drop function corresponding to *N* and applying the first probabilistic drop function to manage the input buffer.

The first range of values may include smaller values of *N* than the second range of values. The method preferably involves computing at least one AQM threshold. A lower AQM threshold may be computed as *N* increases. One or more AQM thresholds may be applied to all active VOQs. Some aspects of the invention involve computing 1st through N^{th} AQM thresholds for 1st through N^{th} classes of service. The method may involve storing at least one of buffer occupancy data, active VOQ data and AQM threshold data.

Some aspects of the method include these steps: receiving a packet; determining an appropriate VOQ for the packet; determining a class of service for the packet; determining a corresponding AQM threshold for the class of service; and determining whether the occupancy of the appropriate VOQ exceeds the

corresponding AQM threshold. An AQM action may be taken when it is determined that the occupancy of the appropriate VOQ exceeds the corresponding AQM threshold. The packet may be added to the appropriate VOQ when it is determined that the occupancy of the appropriate VOQ does not exceed the corresponding AQM threshold.

Alternative embodiments of the invention provide a network device that includes a plurality of ingress line cards, each of which is associated with at least one ingress port, and a plurality of egress line cards, each of which is associated with at least one egress port. Each of the ingress line cards comprises at least one input buffer and is configured to perform the following steps: receive an incoming packet; estimate an overall occupancy level *O* of an input buffer; determine a number *N* of active virtual output queues ("VOQs") of the input buffer; compute a probability *P* based on the overall occupancy level and the number of VOQs; generate a random number *R*; and set a flag when *R* is less than *P*.

Each of the ingress line cards may be configured to perform the following steps: determine an AQM threshold *M* based, at least in part, on *N;* determine a destination VOQ for the packet; and determine whether the destination VOQ contains *M* or more packets. The packet will be placed in the destination VOQ when the destination VOQ contains fewer than *M* packets. An AQM action will be taken when the destination VOQ contains *M* or more packets.

All of the foregoing methods, along with other methods of the present invention, may be implemented by software, firmware and/or hardware. For example, the methods of the present invention may be implemented by computer programs embodied in machine-readable media. Some aspects of the invention can be implemented by network devices or portions thereof, such as individual line cards.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a graph illustrating the RED algorithm.

Fig. 2 is a simplified block diagram of an input-buffered network device.

Fig. 3 illustrates a physical buffer shared by numerous virtual output queues.

Fig. 4 is a flow chart that outlines some methods of the present invention.

Fig. 5 illustrates two subfamilies of probability curves according to some implementations of the invention.

Fig. 6 is one illustrative quantization function that may be used to implement some aspects of the invention.

Fig. 7 is a network device that may be configured to implement some aspects of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

In this application, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be obvious, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps have not been described in detail in order not to obscure the present invention.

The present invention provides AQM methods and devices that are particularly suitable for input-buffered network devices, such as switches and routers. Some aspects of the present invention are particularly suitable for implementing a Low Latency Ethernet ("LLE") solution, also referred to herein as a Data Center Ethernet ("DCE") solution, which simplifies the connectivity of data centers and provides a high bandwidth, low latency network for carrying Ethernet and storage traffic. Some exemplary DCE methods and devices are described in the Cross-Referenced Applications. However, the present invention has wide applicability outside of the DCE context and is generally suitable for any network device, particularly for input-buffered network devices.

According to some aspects of the invention, the probability of setting a "global drop flag" depends, at least in part, on the lesser of a running average of overall buffer occupancy and instantaneous buffer occupancy. Exemplary methods of calculating overall buffer occupancy are described in detail below. As noted elsewhere herein, the global drop flag is not necessarily used to trigger packet drops, but may also be used to trigger other AQM responses, such as explicit congestion notification (e.g., as described in the Cross-Referenced Applications), packet marking, etc.

Preferred implementations of the invention control overall input buffer occupancy while protecting uncongested individual VOQs. Accordingly, preferred implementations of the invention perform the AQM operation only on packets destined for (or belonging to) VOQs that are effectively congested. Some such implementations use a threshold on the instantaneous VOQ length to discriminate between congested and uncongested VOQs. Accordingly, whether an AQM response is made preferably depends on both whether a global drop flag has been set and whether a destination VOQ contains M or more packets, where M is the threshold on the instantaneous VOQ length. Some implementations set different M values for different classes of traffic, e.g., with higher M values for higher-priority traffic.

It has also been observed that when many flows share the same buffer the drop probability should be higher than when just a few flows are active in order to control the average buffer occupancy. Therefore, some aspects of the invention provide an adaptive mechanism to adjust a probability curve (or select another probability curve) based on the number of active VOQs sharing a buffer. In such implementations, the probability of setting a global drop flag depends on the number of active VOQs. AQM responses (e.g., to drop packets) may be taken more aggressively when there is a larger number of VOQs. Therefore, M is preferably smaller when there is a larger number of active VOQs.

Fig. 2 is a simplified block diagram illustrating an input-buffered network device 200 that can be configured in accordance with the present invention. Network device 200 includes ingress line cards 205 and egress line cards 210, interconnected via crossbar 212. In this example, input buffer 217 is receiving packets from source 215 that are being forwarded to uncongested links via port 220. Input buffer 217 is also receiving packets from sources 225, 230 and 235 that are being forwarded to

uncongested links via other ports, including port 240. It is highly desirable that the flows to uncongested links are not inordinately slowed by the fact that flows to congested links are being processed by the same input buffer 217.

Fig. 3 is a schematic diagram that illustrates multiple VOQs 305 within input buffer 217. As known by those of skill in the art, packets for each output destination will be assigned to a corresponding VOQ, each of which may include packets from more than one flow. In other words, the packets are organized into VOQs according to their output destination, regardless of where the packets originate.

Fig. 3 also illustrates global drop thresholds 310, 320 and 330. As will be discussed in more detail below, these drop thresholds indicate maximum numbers of packets *M* that may be in a VOQ before an AQM action is taken under certain conditions. Arrow 340 is intended to represent the instantaneous buffer occupancy at a particular time.

An overview of some implementations of the invention will now be described with reference to the flow chart of Fig. 4. It will be appreciated by those of skill in the art that the steps illustrated and described herein are not necessarily performed in the order indicated. Moreover, it will be appreciated that some aspects of the invention may be performed using more (or fewer) steps than are indicated herein.

In step 401, a packet arrives at an input buffer. In step 403, the number of "active" VOQs, meaning the number of VOQs that have packets, is determined. As will be described in more detail below, the number of active VOQs may determine (at least in part) the value of an AQM threshold *M*, also referred to herein as a "drop threshold." In step 405 the global drop flag is evaluated: if the flag is set and the length of the VOQ the packet is going to be added to is larger than a drop threshold *M*, step 440 (described later) is taken. Otherwise, the overall average buffer occupancy is then determined. (Step 410.) In some implementations, step 403 and/or step 410 is not made in response to a packet's arrival, but instead the determination is made upon the occurrence of another event or at a fixed time interval.

In some implementations of the invention, at least step 410 (and possibly other steps, such as step 403) is performed at fixed time intervals *T*. *T* may be set to any convenient time, for example, a time that is comparable with the transmission time of

an MTU frame on the egress interface. For example, T would be 1.2 μs if a 1500-byte frame were transmitted at 10 Gbps.

Performing such calculations at fixed time intervals can be advantageous as compared to having the computations triggered by a packet's arrival. For example, when an idle period follows a busy period and these computations are triggered by a packet's arrival, the overall buffer occupancy calculation is not updated until the first packet arrival after the idle period. In this case, the calculated overall buffer occupancy could remain high even if the instantaneous queue size were actually close to or at zero. Therefore, if the average queue size were still larger than the minimum threshold, packets would continue to be discarded even if the queue were empty. This would cause lower throughput because the link would become underutilized.

Some implementations of the invention compute an overall average buffer occupancy `AVG_BUF_LEN`, which is calculated according to the following exponential weighted moving average:

$$\texttt{AVG\_BUF\_LEN} = \texttt{AVG\_BUF\_LEN}_{\texttt{OLD}} * (1 - \texttt{W}) + \texttt{INST\_BUF\_LEN} * \texttt{W}$$

(Equation 1)

In Equation (1), `INST_BUF_LEN` is the instantaneous buffer occupancy and `W` is a weight that determines the sensitivity of `AVG_BUF_LEN` to the instantaneous fluctuations in the buffer occupancy. In some preferred implementations, `W` is a relatively small number (e.g., in the range of .001 to .1), such that `AVG_BUF_LEN` has a relatively low sensitivity to instantaneous fluctuations in the buffer occupancy.

However, some implementations of the invention provide for a sensitivity to instantaneous decreases in the buffer occupancy. Preferably, the calculated value for the overall buffer occupancy should be quickly reduced when the instantaneous queue size decreases rapidly. Otherwise, packets could unnecessarily be discarded, e.g., as explained above. Therefore, some implementations of the invention consider the overall buffer occupancy to be the lesser of a calculated rolling average buffer occupancy and an instantaneous buffer occupancy. In some such implementations, at any given time the computed buffer occupancy (`COMP_BUF_LEN`) is defined as the minimum of the average and the instantaneous buffer occupancy:

COMP_BUF_LEN = min(AVG_BUF_LEN, INST_BUF_LEN) (Equation 2)

In step 415, the probability of setting a global drop flag is calculated. This probability is preferably a function of the computed overall buffer occupancy, e.g., of COMP_BUF_LEN. However, some aspects of the present invention provide more sophisticated methods of calculating such a probability. Some such aspects of the invention have been implemented in response to the observation that packets need to be dropped more aggressively when many flows share the same buffer. To implement such a behavior, the probability has been turned into a function of two variables, i.e., the computed buffer occupancy (COMP_BUF_LEN) determined in step 410 and the number of active VOQs (N_ACT_VOQ) determined in step 405.

Accordingly, in some implementations of the invention, for every packet arrival a drop probability p is calculated as follows:

p = P(COMP_BUF_LEN, N_ACT_VOQ) (Equation 3)

In Equation 3, P(,) is a probability function. Although many such probability functions are within the scope of the invention, Fig. 5 illustrates a probability function 500 according to one such aspect of the invention. The family of curves shown in Fig. 5 is composed of two subfamilies, subfamily 510 and subfamily 520. Subfamily 510 is associated with exponential functions of the following kind:

$$p = k\,(f(\texttt{N\_ACT\_VOQ})^{\,g(\texttt{COMP\_BUF\_LEN})} - 1) \quad \text{(Equation 4)}$$

In Equation 4, k is a constant scaling factor, f is a function that returns a base for the exponentiation dependent on the number of active VOQs and g is a function that returns an exponent dependent on the current computed average buffer occupancy. The functions f and g may be simple linear functions or more complex non-linear functions, used for example to emphasize the precision on the calculation of the probability for small values of N_ACT_VOQ and COMP_BUF_LEN.

Subfamily 520 is associated with logarithmic functions of the following kind:

$$p = \log_{f(\texttt{N\_ACT\_VOQ})}(g\,(\texttt{COMP\_BUF\_LEN}) - 1) \quad \text{(Equation 5)}$$

As indicated by arrow 530, when the number of active VOQs increases beyond a predetermined range, the probability function is switched from exponential subfamily 510 to logarithmic subfamily 520. The rationale in doing so is based on the fact that when a few flows share the buffer, it is important to keep the probability of dropping packets very low when the buffer occupancy is low, but also to increase the probability rapidly as the buffer occupancy increases. Accordingly, for small values of `N_ACT_VOQ`, a probability function is appropriate that increases less rapidly than linear function 550 for small values of `COMP_BUF_LEN` and that increases more rapidly than linear function 550 for larger values of `COMP_BUF_LEN` (with an overall concave-upwards form).

Conversely, when many flows share the buffer, it is preferable to start dropping packets aggressively even with a low buffer occupancy. Therefore, for larger values of `N_ACT_VOQ`, a probability function is appropriate that increases more rapidly than linear function 550, even for small values of `COMP_BUF_LEN` (with an overall concave-downwards form). It has been observed that if a gentler probability function is used with many flows, it is very likely that a substantial portion of them will be unaffected by drops or other AQM responses. Consequently, these flows may be allowed to open their window enough to saturate the buffer in case of TCP flows, or cause buffer overflows because not enough packets are dropped in case of traffic via other protocols.

Referring again to Fig. 4, in step 420 a random number `r` in the range [0, 1] is generated. The random number is then compared to `p`. (Step 425.) If `r > p`, the packet is included in the VOQ for its indicated destination. (Step 445.)

However, if `r =< p`, a global drop flag is set. (Step 430.) A "drop flag" indicates that a conditional decision has been made to implement an AQM response. The AQM response could be, for example, a head of line packet drop, a tail packet drop and/or another AQM response such as an explicit congestion notification, e.g., as described in the Cross-Referenced Applications.

In step 435, at least one AQM threshold M is checked. An AQM threshold indicates a maximum number of packets that may be stored in a VOQ before the application of an AQM action, such as dropping a packet. In the "tail drop" example

of method 400, it is determined in step 435 whether the incoming packet is destined for a VOQ having a number of packets that is greater than *M*. If the indicated VOQ has M or fewer packets, the packet is placed in the VOQ. (Step 445.) If the indicated VOQ has more than M packets, an AQM action is taken (e.g., the packet is dropped) and the global drop flag is reset (step 440).

The invention may also be applied to implementations involving the dropping of packets from the head of a VOQ. Instead of determining whether to discard or enqueue an incoming packet, a decision can be made regarding the disposition of a packet at the head of a VOQ.

The rationale behind this behavior is that if a packet is scheduled (or would be scheduled) for transmission from a "long" VOQ (based on its length exceeding drop threshold *M*), it is very likely that such VOQ is congested and it is therefore contributing to the overall buffer congestion. Thus, it is appropriate to drop the next packet coming out of such VOQ (or the next packet that would have been placed in the VOQ). Conversely, if the VOQ is nearly empty, there is no point in dropping a packet going to (or coming from) that VOQ, as the traffic therein stored is unlikely to be a contributor to the buffer congestion.

In some implementations, a relatively low `N_ACT_VOQ` will result in a relatively high value of *M*. For example, if `N_ACT_VOQ` is low, *M* could be set to level 310 of Fig. 3. Conversely, a relatively high `N_ACT_VOQ` will result in a relatively low value of *M*. For example, if `N_ACT_VOQ` is high, *M* could be set to level 330 of Fig. 3.

Some implementations of the invention involve the establishment of multiple drop thresholds M_1 through M_N for a single value of `N_ACT_VOQ`. In some such implementations, if the priority and/or service class of the packet is relatively higher, then *M* will be relatively larger. For example, thresholds 310, 320 and 330 of Fig. 3 could all be used for a single value of `N_ACT_VOQ`. Threshold 310 could be used for packets with a relatively high service class or priority and threshold 330 could be used for packets with a relatively low service class or priority. By having different thresholds on a per priority basis, a different drop priority or weight is provided for, e.g., different classes of traffic, for control frames vs. data frames, etc.

The algorithms described herein are sufficiently simple to be implemented in hardware in a straightforward way. In some preferred implementations, the probability function `P(,)` is implemented with a lookup table, because performing the calculations for Equations 4 and 5 in hardware would require very fast processing (e.g., a very fast FPU). However, in some cases a lookup table might not be feasible. Moreover, the table for drop thresholds may use considerable space, because it is to be indexed by the number of active VOQs.

Therefore, a compression of the `COMP_BUF_LEN` and `N_ACT_VOQ` values can be advantageous in order to save memory space. The simplest form of compression can be achieved by linear quantization, i.e., by removing a certain number of LSBs from the aforementioned values.

Some extra space can be saved if a more sophisticated non-linear quantization technique is employed for compressing data, e.g. for compressing the values of `COMP_BUF_LEN`, `N_ACT_VOQ` and/or corresponding AQM threshold data. One such non-linear quantization technique will now be described with reference to Fig. 6. The vertical axis 605 of graph 600 indicates the number of individual probability curves and horizontal axis 610 indicates numbers of active VOQs. Here, curves 615 correspond with the logarithmic subfamily of curves 520 of Fig. 5 and curves 620 correspond with exponential subfamily 510. For example, by using the non-linear quantization function shown in Fig. 6, which maps 8192 values on 16 values, the size of the probability table will be 16 x 16 x 8 bits = 2048 bit.

A hybrid approach may also be employed by, for example, using an 8-bit linearly quantized value for `COMP_BUF_LEN` and a 4-bit non-linearly quantized value for `N_ACT_VOQ`. In this example, the probability table would be as large 256 x 16 x 8 = 32 Kbit, and the threshold table would be as large as 16 x 13 x 3 = 624 bit. This exemplary non-linear quantization function can be implemented, for example, with a lookup table 8192 x 4 = 32 Kbit large or as a battery of 15 13-bit comparators plus an encoder, if "silicon real estate" is an issue.

Fig. 7 illustrates an example of a network device that may be configured to implement some methods of the present invention. Network device 760 includes a master central processing unit (CPU) 762, interfaces 768, and a bus 767 (e.g., a PCI

bus). Generally, interfaces 768 include ports 769 appropriate for communication with the appropriate media. In some embodiments, one or more of interfaces 768 includes at least one independent processor 774 and, in some instances, volatile RAM. Independent processors 774 may be, for example ASICs or any other appropriate processors. According to some such embodiments, these independent processors 774 perform at least some of the functions of the logic described herein. In some embodiments, one or more of interfaces 768 control such communications-intensive tasks as media control and management. By providing separate processors for the communications-intensive tasks, interfaces 768 allow the master microprocessor 762 efficiently to perform other functions such as routing computations, network diagnostics, security functions, etc.

The interfaces 768 are typically provided as interface cards (sometimes referred to as "line cards"). Generally, interfaces 768 control the sending and receiving of data packets over the network and sometimes support other peripherals used with the network device 760. Among the interfaces that may be provided are Fibre Channel ("FC") interfaces, Ethernet interfaces, frame relay interfaces, cable interfaces, DSL interfaces, token ring interfaces, and the like. In addition, various very high-speed interfaces may be provided, such as fast Ethernet interfaces, Gigabit Ethernet interfaces, ATM interfaces, HSSI interfaces, POS interfaces, FDDI interfaces, ASI interfaces, DHEI interfaces and the like.

When acting under the control of appropriate software or firmware, in some implementations of the invention CPU 762 may be responsible for implementing specific functions associated with the functions of a desired network device. According to some embodiments, CPU 762 accomplishes all these functions under the control of software including an operating system (e.g. Linux, VxWorks, etc.), and any appropriate applications software.

CPU 762 may include one or more processors 763 such as a processor from the Motorola family of microprocessors or the MIPS family of microprocessors. In an alternative embodiment, processor 763 is specially designed hardware for controlling the operations of network device 760. In a specific embodiment, a memory 761 (such as non-volatile RAM and/or ROM) also forms part of CPU 762. However, there are many different ways in which memory could be coupled to the system. Memory

block 761 may be used for a variety of purposes such as, for example, caching and/or storing data, programming instructions, etc.

Regardless of network device's configuration, it may employ one or more memories or memory modules (such as, for example, memory block 765) configured to store data, program instructions for the general-purpose network operations and/or other information relating to the functionality of the techniques described herein. The program instructions may control the operation of an operating system and/or one or more applications, for example.

Because such information and program instructions may be employed to implement the systems/methods described herein, the present invention relates to machine-readable media that include program instructions, state information, etc. for performing various operations described herein. Examples of machine-readable media include, but are not limited to, magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks; magneto-optical media; and hardware devices that are specially configured to store and perform program instructions, such as read-only memory devices (ROM) and random access memory (RAM). The invention may also be embodied in a carrier wave traveling over an appropriate medium such as airwaves, optical lines, electric lines, etc. Examples of program instructions include both machine code, such as produced by a compiler, and files containing higher level code that may be executed by the computer using an interpreter.

Although the system shown in Fig. 7 illustrates one specific network device of the present invention, it is by no means the only network device architecture on which the present invention can be implemented. For example, an architecture having a single processor that handles communications as well as routing computations, etc. is often used. Further, other types of interfaces and media could also be used with the network device. The communication path between interfaces/line cards may be bus based (as shown in Fig. 7) or switch fabric based (such as a cross-bar).

Other Embodiments

Although illustrative embodiments and applications of this invention are shown and described herein, many variations and modifications are possible which

remain within the concept, scope, and spirit of the invention, and these variations would become clear to those of ordinary skill in the art after perusal of this application.

Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

WE CLAIM:

1. An active queue management ("AQM") method, comprising:
receiving an incoming packet;
estimating an overall occupancy level *O* of an input buffer;
determining a number *N* of active virtual output queues ("VOQs") of the input buffer;
computing a probability *P* based on the overall occupancy level and the number of VOQs;
generating a random number *R*; and
setting a flag when *R* is less than *P*.

2. The method of claim 1, wherein *P* increases when *N* increases.

3. The method of claim 1, wherein *O* is a running average occupancy level of the input buffer.

4. The method of claim 1, wherein *O* is the lesser of *R* and *I,* wherein *R* is a running average occupancy level of the input buffer and *I* is an instantaneous occupancy level of the input buffer.

5. The method of claim 1, further comprising:
determining an AQM threshold *M* based, at least in part, on *N;*
determining a destination VOQ for the packet; and
determining whether the destination VOQ contains *M* or more packets.

6. The method of claim 1, further comprising:
determining a plurality of AQM thresholds, each AQM threshold being based at least in part on a packet's class of service,
determining a class of service of the packet;
determining a destination VOQ for the packet; and
determining whether the destination VOQ contains more packets than the AQM threshold for the determined class of service.

7. The method of claim 5, further comprising the step of forwarding the packet to the destination VOQ when the destination VOQ contains fewer than *M* packets.

8. The method of claim 5, further comprising the step of taking an AQM action when the destination VOQ contains *M* or more packets.

9. The method of claim 5, wherein *M* decreases as the number of active VOQs increases.

10. The method of claim 6, further comprising the step of taking an AQM action when the destination VOQ contains more packets than the AQM threshold for the determined class of service.

11. The method of claim 7, wherein the AQM action comprises at least one of dropping the packet, dropping a packet at the head of the destination VOQ, marking the packet and sending an explicit congestion notification.

12. An active queue management ("AQM") method, comprising:

receiving an incoming packet;

estimating an overall occupancy level *O* of an input buffer;

computing a probability *P* based at least in part on the overall occupancy level;

generating a random number *R*;

setting a flag if *R* is less than *P*;

determining an AQM threshold *M;* and

determining whether a destination virtual output queue ("VOQ") of the packet contains *M* or more packets.

13. The method of claim 12, further comprising the step of sending the packet to the destination VOQ when the destination VOQ contains fewer than *M* packets.

14. The method of claim 12, further comprising the step of taking an AQM action when the destination VOQ contains *M* or more packets.

15. The method of claim 12, wherein *P* is also based in part upon a number of active VOQs.

16. The method of claim 12, wherein *O* is a running average occupancy level of the input buffer.

17. The method of claim 12, wherein *O* is the lesser of *R* and *I,* wherein *R* is a running average occupancy level of the input buffer and *I* is an instantaneous occupancy level of the input buffer.

18. The method of claim 12, wherein *M* decreases as the number of active VOQs increases.

19. The method of claim 14, wherein the AQM action comprises at least one of dropping the packet, dropping a packet at the head of the destination VOQ, marking the packet and sending an explicit congestion notification.

20. The method of claim 15, wherein *P* increases when the number of VOQs increases.

21. A method of managing input buffers in a network device, the method comprising:

determining a first range of values of *N,* where *N* is the number of active virtual output queues (VOQs) in an input buffer;

determining a second range of values of *N;*

assigning a first family of probabilistic drop functions to the first range of values, the first family of probabilistic drop functions being exponential functions of *N* and computed buffer occupancy *C;* and

assigning a second family of probabilistic drop functions to the second range of values, the second family of probabilistic drop functions being logarithmic functions of *N* and *C*;

determining *N* at a first time;

determining a first probabilistic drop function corresponding to *N;* and

applying the first probabilistic drop function to manage the input buffer.

22. The method of claim 21, wherein the first range of values comprises smaller values of N than the second range of values.

23. The method of claim 21, further comprising the step of computing at least one active queue management ("AQM") threshold.

24. The method of claim 23, wherein the computing step comprises computing a lower AQM threshold as N increases.

25. The method of claim 23, further comprising the step of applying at least one AQM threshold to all active VOQs.

26. The method of claim 23, further comprising the step of computing 1st through N^{th} AQM thresholds for 1st through N^{th} classes of service.

27. The method of claim 26, further comprising:
receiving a packet;
determining an appropriate VOQ for the packet;
determining a class of service for the packet;
determining a corresponding AQM threshold for the class of service; and
determining whether the occupancy of the appropriate VOQ exceeds the corresponding AQM threshold.

28. The method of claim 26, further comprising the step of storing at least one of buffer occupancy data, active VOQ data and AQM threshold data.

29. The method of claim 27, further comprising the step of taking an AQM action when it is determined that the occupancy of the appropriate VOQ exceeds the corresponding AQM threshold.

30. The method of claim 27, further comprising the step of adding the packet to the appropriate VOQ when it is determined that the occupancy of the appropriate VOQ does not exceed the corresponding AQM threshold.

31. A network device, comprising:

a plurality of ingress line cards, each line card associated with at least one ingress port; and

a plurality of egress line cards, each line card associated with at least one egress port;

wherein each of the plurality of ingress line cards comprises at least one input buffer and is configured to perform the following steps:

receive an incoming packet;

estimate an overall occupancy level *O* of an input buffer;

determine a number *N* of active virtual output queues ("VOQs") of the input buffer;

compute a probability *P* based on the overall occupancy level and the number of VOQs;

generate a random number *R*; and

set a flag when *R* is less than *P*.

32. The network device of claim 31, wherein each of the plurality of ingress line cards is further configured to perform the following steps:

determine an AQM threshold *M* based, at least in part, on *N;*

determine a destination VOQ for the packet; and

determine whether the destination VOQ contains *M* or more packets.

33. The network device of claim 32, wherein each of the plurality of ingress line cards is further configured to forward the packet to the destination VOQ when the destination VOQ contains fewer than *M* packets.

34. The network device of claim 32, wherein each of the plurality of ingress line cards is further configured to take an AQM action when the destination VOQ contains *M* or more packets.

35. A network device, comprising:

means for receiving an incoming packet;

means for estimating an overall occupancy level *O* of an input buffer;

means for computing a probability *P* based at least in part on the overall occupancy level;

means for generating a random number *R*;

means for setting a flag if *R* is less than *P*;

means for determining an AQM threshold *M;* and

means for determining whether a destination virtual output queue ("VOQ") of the packet contains *M* or more packets.

36. The network device of claim 35, further comprising means for sending the packet to the destination VOQ when the destination VOQ contains fewer than *M* packets.

37. The network device of claim 35, further comprising means for taking an AQM action when the destination VOQ contains *M* or more packets.

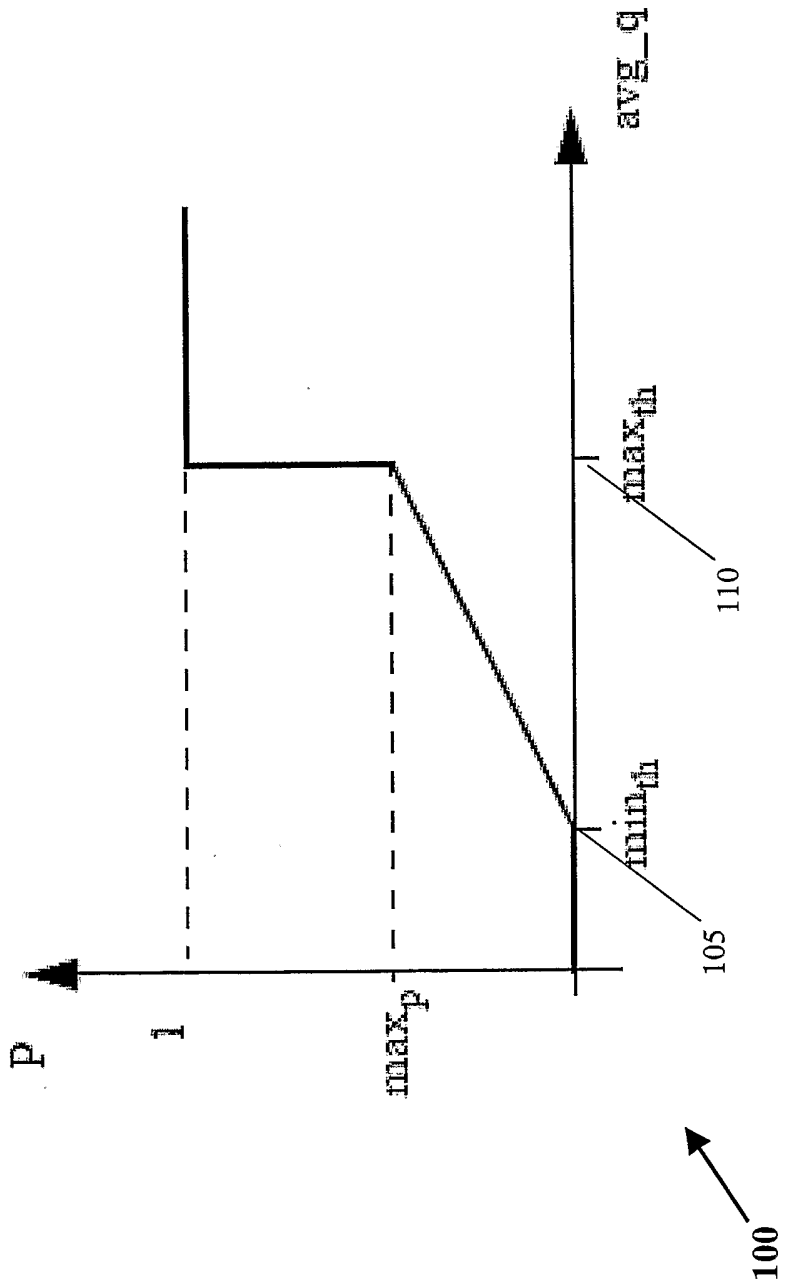


Fig. 1

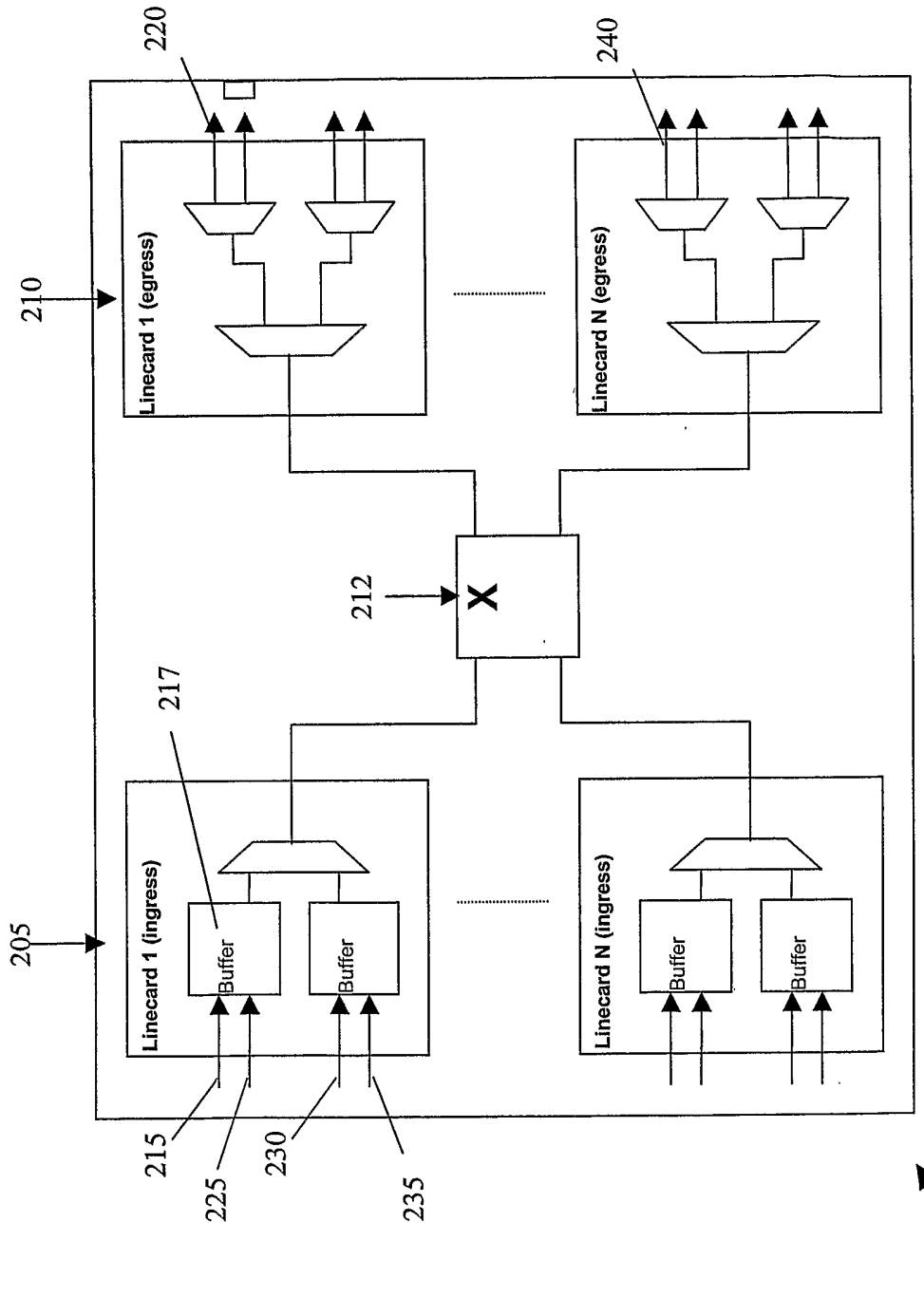

220
240
210
Linecard 1 (egress)
Linecard N (egress)
212
X
217
205
Linecard 1 (ingress)
Linecard N (ingress)
Buffer
Buffer
Buffer
Buffer
215
225
230
235
200

Fig. 2

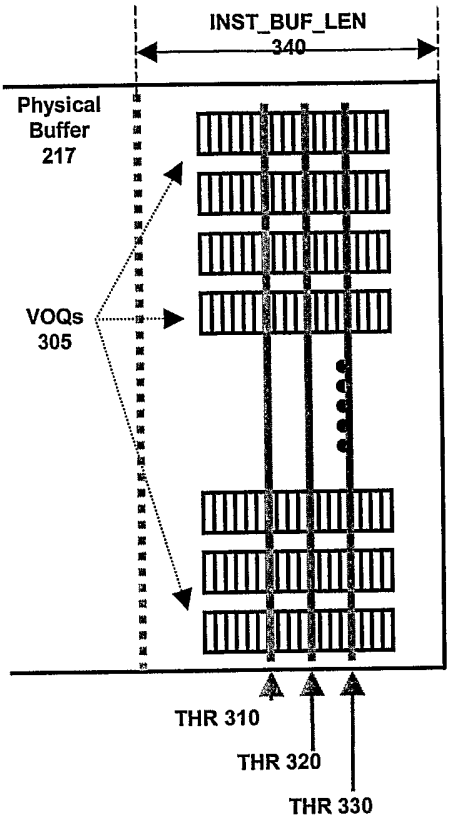

INST_BUF_LEN
340
Physical
Buffer
217
VOQs
305
THR 310
THR 320
THR 330

Fig. 3

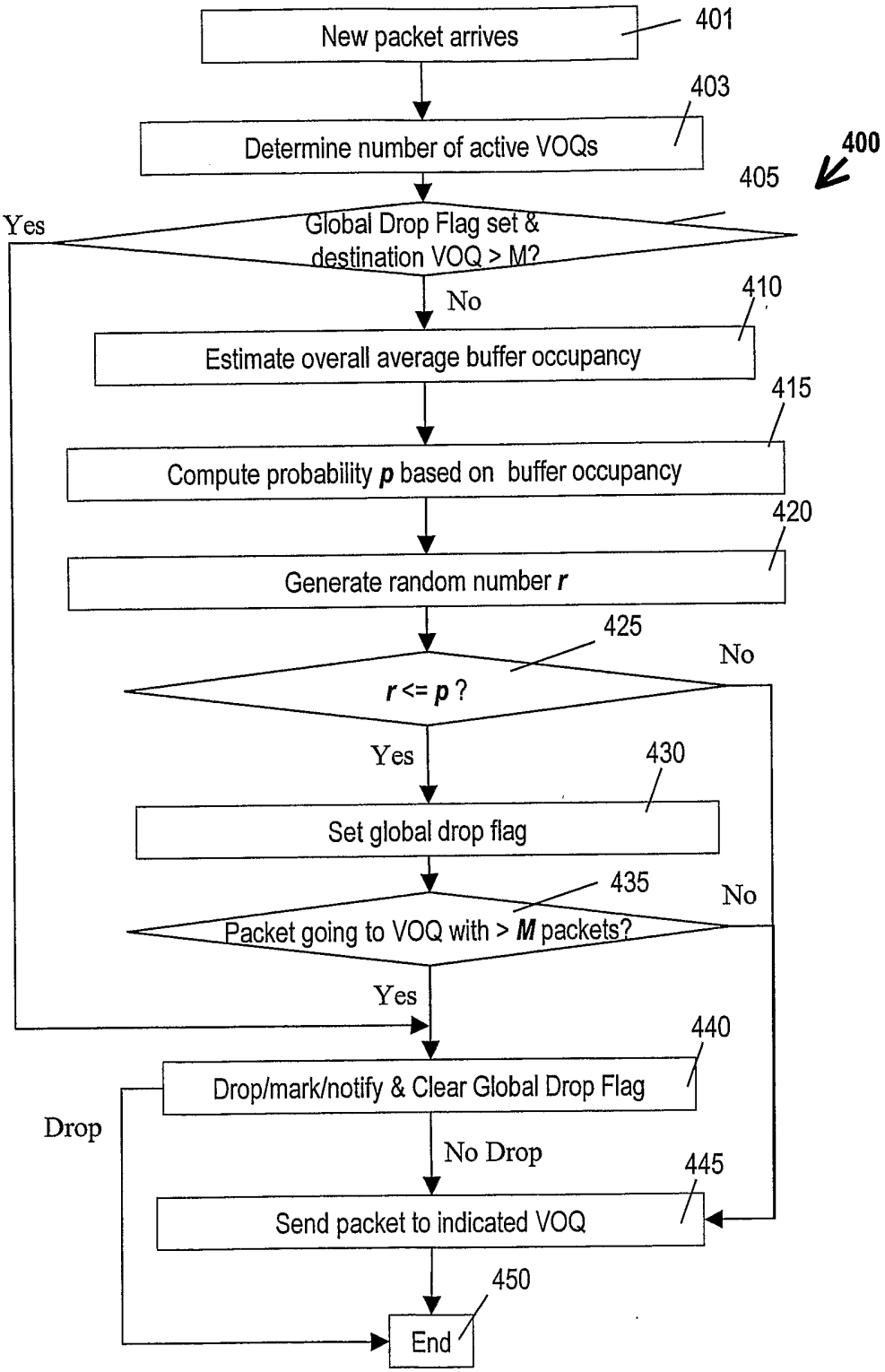

New packet arrives
401
Determine number of active VOQs
403
400
405
Yes
Global Drop Flag set & destination VOQ > M?
No
410
Estimate overall average buffer occupancy
415
Compute probability p based on buffer occupancy
420
Generate random number r
425
No
r <= p ?
Yes
430
Set global drop flag
435
No
Packet going to VOQ with > M packets?
Yes
440
Drop/mark/notify & Clear Global Drop Flag
Drop
No Drop
445
Send packet to indicated VOQ
450
End

Fig. 4

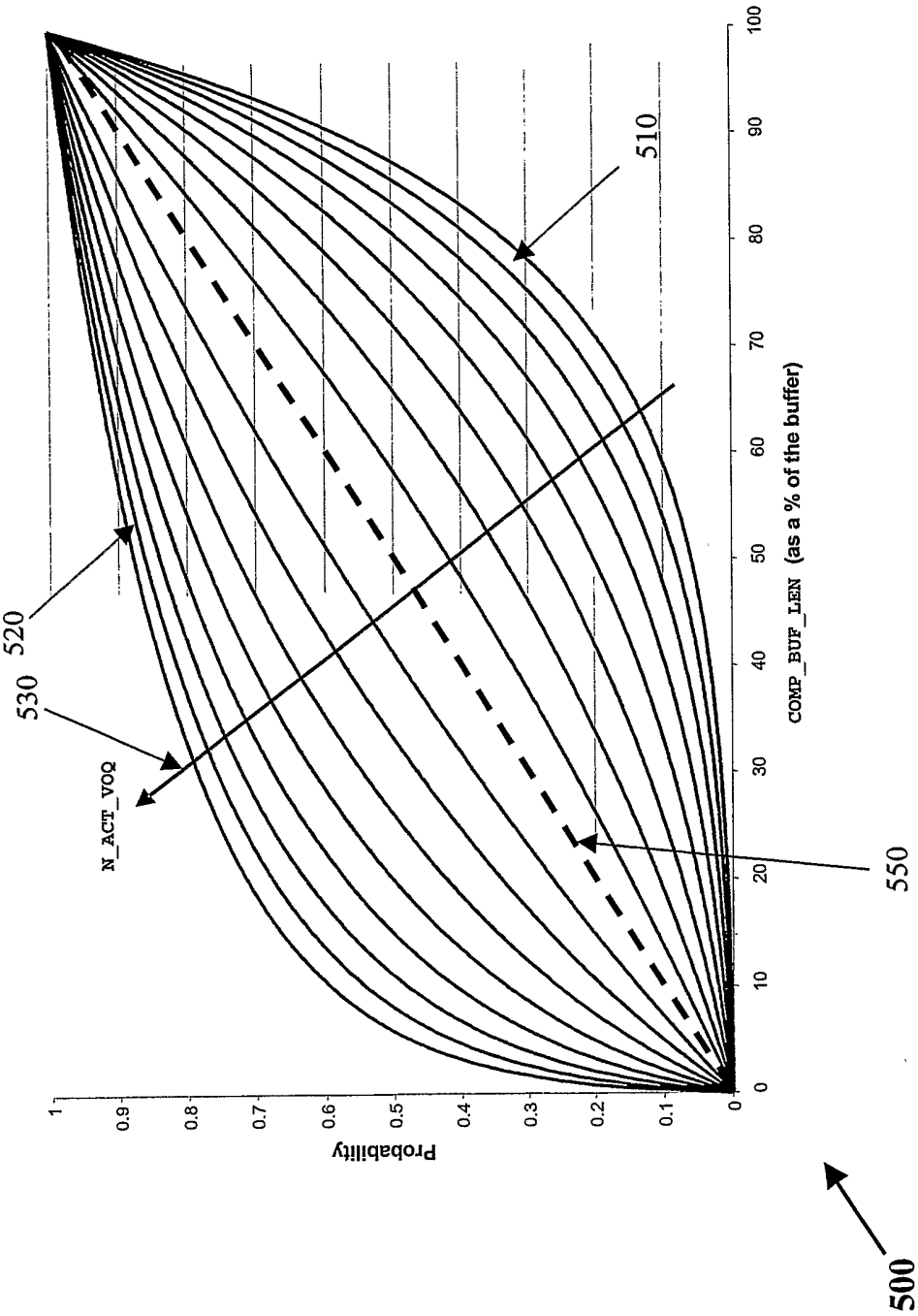

500
510
520
530
550
Probability
COMP_BUF_LEN (as a % of the buffer)
N_ACT_VOQ
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10
20
30
40
50
60
70
80
90
100

Fig. 5

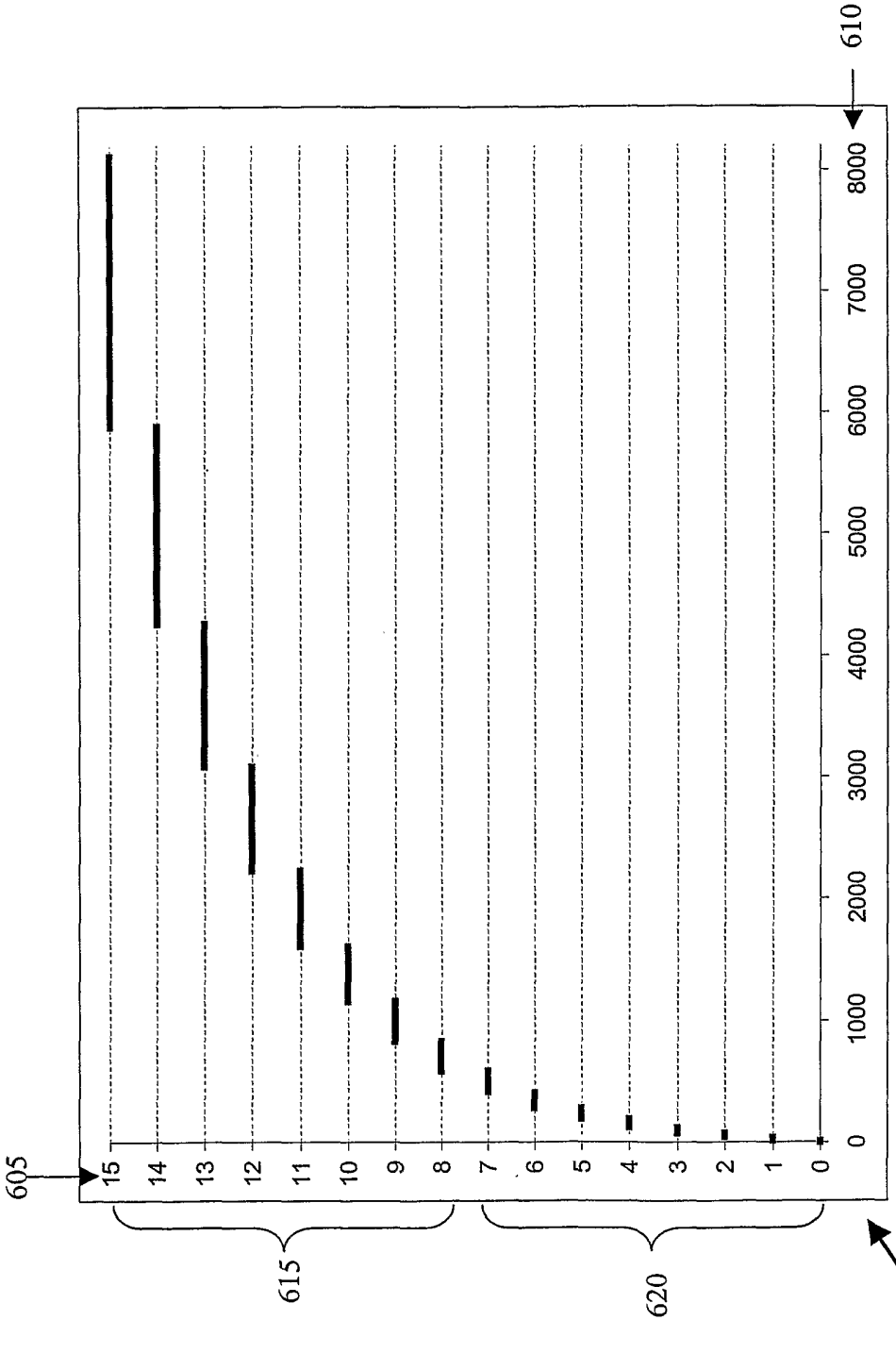

610
605
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
0
1000
2000
3000
4000
5000
6000
7000
8000
615
620
600

Fig. 6

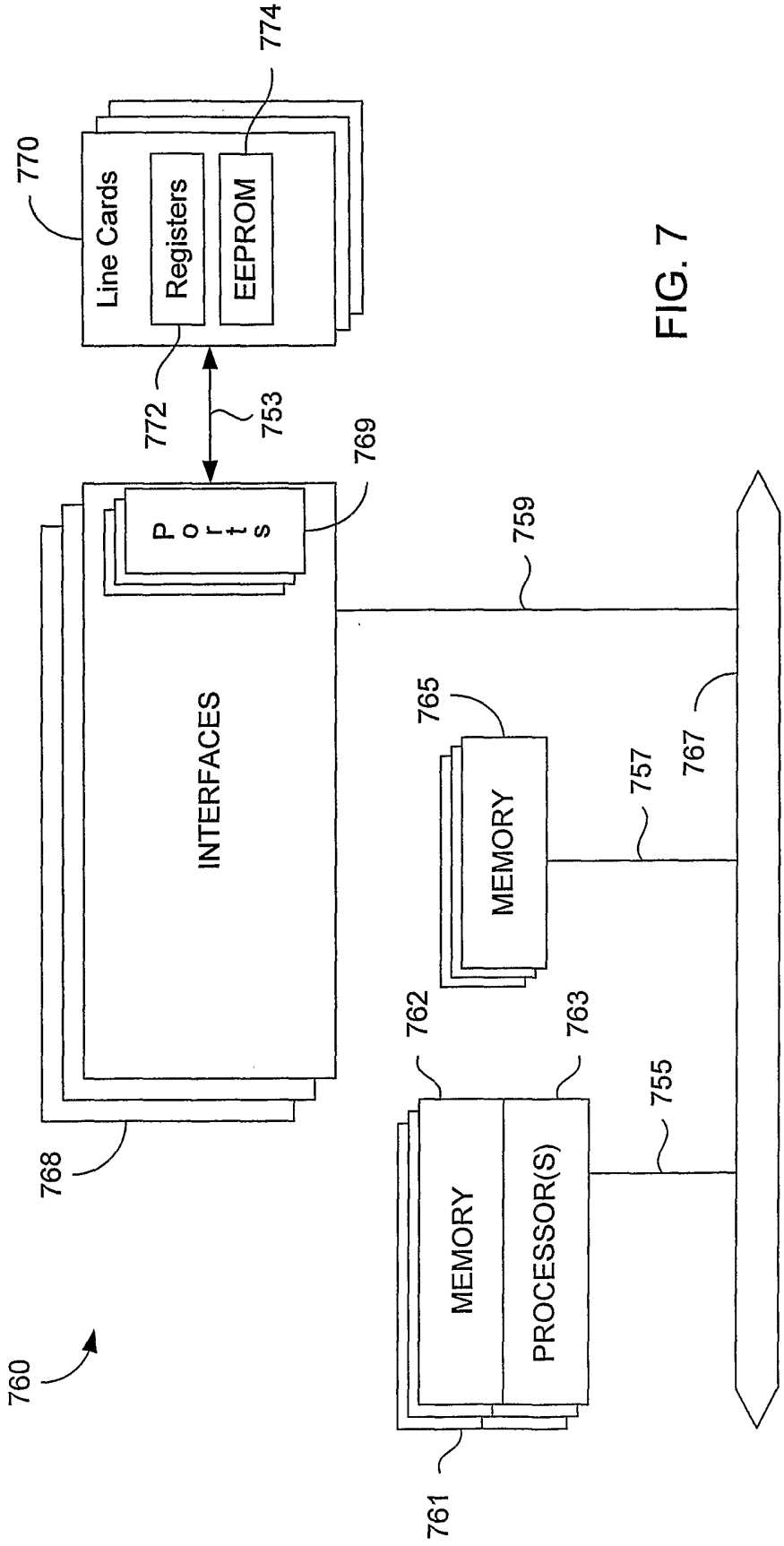

760
761
762
763
765
767
768
769
770
772
774
753
755
757
759
MEMORY
PROCESSOR(S)
MEMORY
INTERFACES
Ports
Line Cards
Registers
EEPROM

FIG. 7